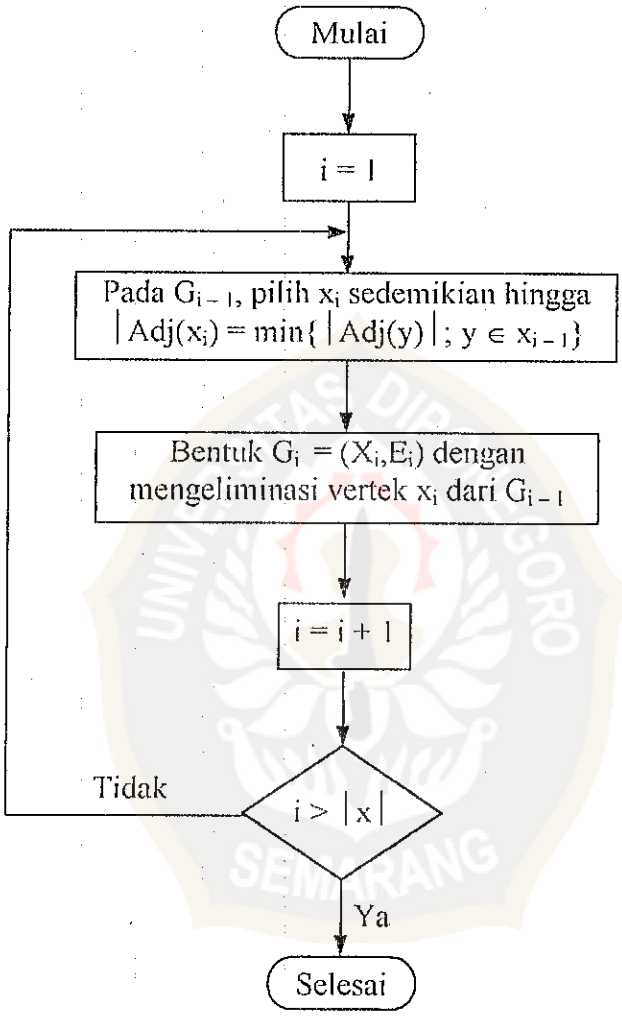
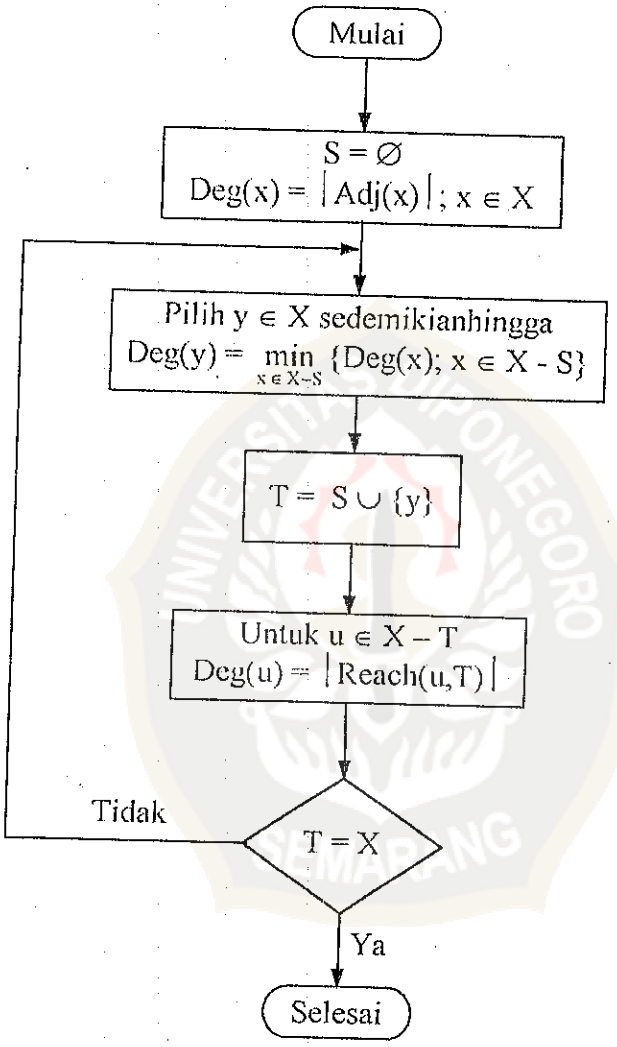
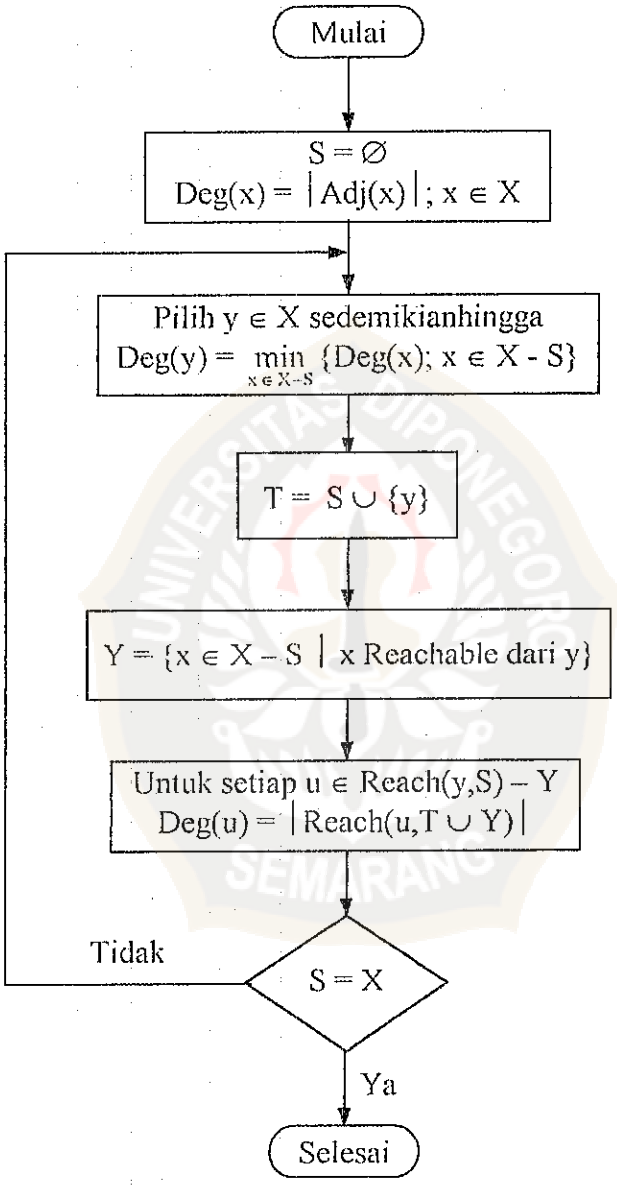
Lampiran 1.1. Diagram Alir Algoritma Derajat Minimum menggunakan Eliminasi Graph.



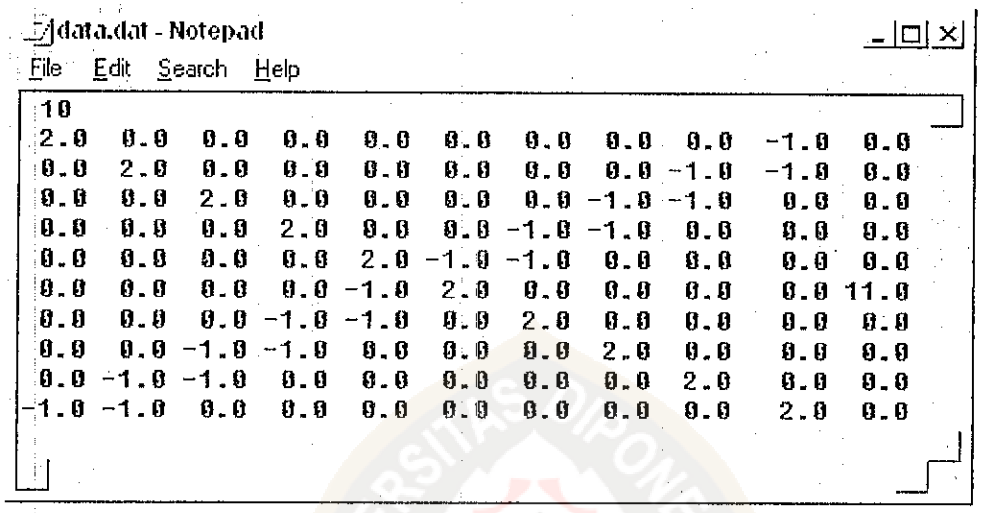
Lampiran 1.2. Diagram Alir Algoritma Derajat Minimum menggunakan Reachable Set.



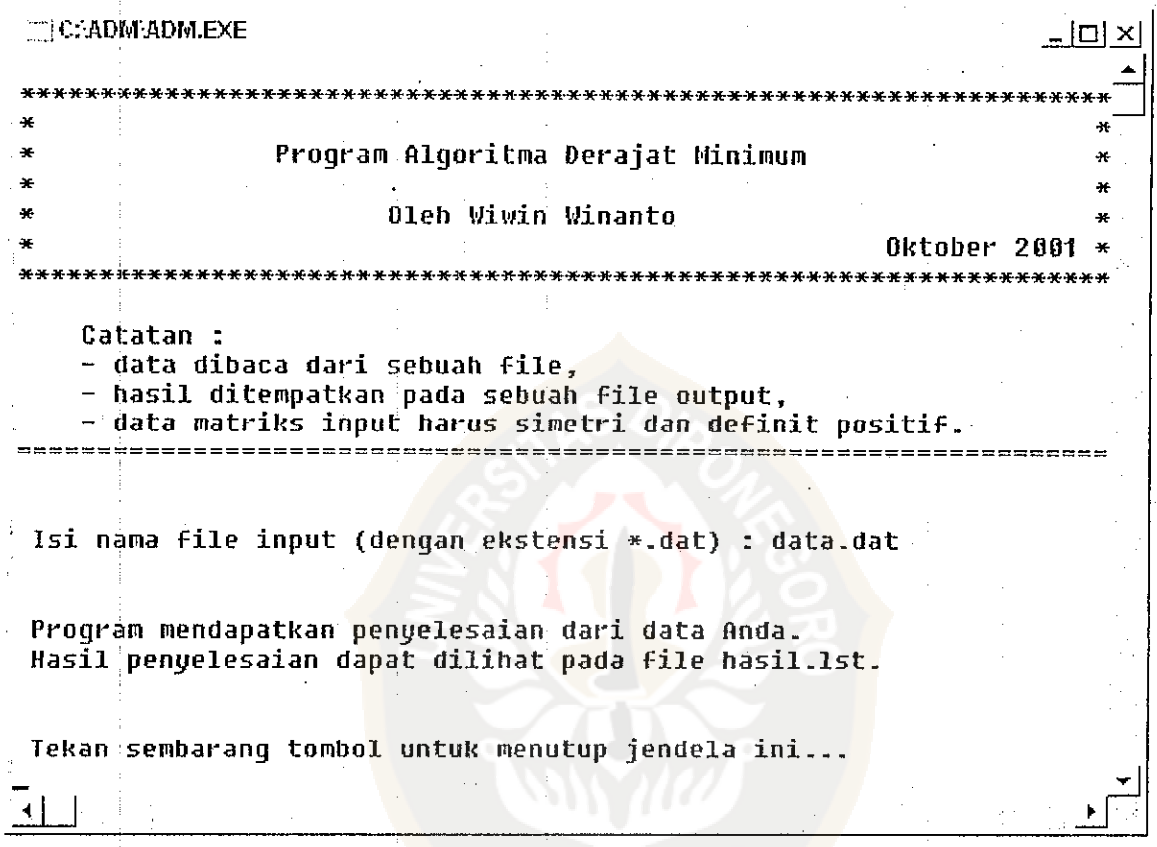
Lampiran 1.3. Diagram Alir Algoritma Derajat Minimum setelah Perbaikan.



Lampiran 1.4. Format Input Data Matriks.



Lampiran 1.5. Tampilan Input Data.



Tampilan Output dari Program.

```
**********************************************************************
*                                                                    *
*                  Program Algoritma Derajat Minimum                 *
*                                                                    *
*                        Oleh Wiwin Winanto                          *
*                                                    Oktober 2001    *
**********************************************************************
```

Sistem linier yang akan diselesaikan :

N = 10

2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000
0.000	2.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	-1.000	0.000
0.000	0.000	2.000	0.000	0.000	0.000	0.000	-1.000	-1.000	0.000	0.000
0.000	0.000	0.000	2.000	0.000	0.000	-1.000	-1.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	2.000	-1.000	-1.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	-1.000	2.000	0.000	0.000	0.000	0.000	11.000
0.000	0.000	0.000	-1.000	-1.000	0.000	2.000	0.000	0.000	0.000	0.000
0.000	0.000	-1.000	-1.000	0.000	0.000	0.000	2.000	0.000	0.000	0.000
0.000	-1.000	-1.000	0.000	0.000	0.000	0.000	0.000	2.000	0.000	0.000
-1.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2.000	0.000

Penyelesaian sistem persamaan linier menggunakan Algoritma Derajat Minimum.

Jumlah persamaan N = 10

Ada NADJ = 18 adjacency.

Tampilan struktur adjacency dari matriks sparse.
Total ada 300 entri.

Baris	Tidak nol	
1	10	
2	9	10
3	8	9
4	7	8
5	6	7
6	5	
7	4	5
8	3	4
9	2	3
10	1	2

Tampilan struktur tidak nol dari matriks.

```
 1 X      X
 2 X     XX
 3  X   XX
 4   X XX
 5    XXX
 6    XX
 7   XX X
 8  XX  X
 9 XX    X
10 XX     X
```

Solusi :

1.000000	3.000000	5.000000	7.000000	9.000000
10.000000	8.000000	6.000000	4.000000	2.000000

ADM
Hasil yang benar.

Lampiran 1.7. Program Algoritma Derajat Minimum dengan Pascal 1.5 for Windows.

```
{************************************************************ *
 *                                                            *
 *                Program Algoritma Derajat Minimum           *
 *                                                            *
 *                Oleh : Wiwin Winanto                        *
 * ---------------------------------------------------------- *
 *                                                 Oktober 2001 *
 ************************************************************
   =                                                        =
   = Catatan :                                              =
   =                                                        =
   = - data dibaca dari sebuah file,                        =
   = - hasil ditempatkan pada sebuah file output,           =
   = - data matriks input harus simetri dan definit positif. =
 ************************************************************}

Program ADM;

Uses WinCrt;
Label fin;
Const
  maxadj       = 300;
  maxenv       = 300;
  maxnzsub     = 300;
  size         = 100;
Type
  pVmadj       = ^Vmaxadj;
  Vmaxadj      = Array[1..maxadj] of INTEGER;
  pVn          = ^Vn;
  Vn           = Array[1..size] of REAL;
  pVenv        = ^Vmaxenv;
  Vmaxenv      = Array[1..maxenv] of REAL;
  pVin         = ^Vin;
  Vin          = Array[1..size] of INTEGER;
  pVinp        = ^Vinp;
  Vinp         = Array[1..size+1] of INTEGER;
  pVi99        = ^Vi99;
  Vi99         = Array[1..99] of INTEGER;
  pnzsub       = ^Vmaxnzsub;
  Vmaxnzsub    = Array[1..maxnzsub] of INTEGER;
  pTlznr       = ^Tlznr;
  Tlznr        = Array[1..99] of REAL;
  pMat         = ^Mat;
  Mat          = Array[1..size,1..size] of REAL;

Var
  name: string[40];
```

```
  fp1,fp2 : TEXT;
  lnzr: pTlznr  ;
  n: Integer;
  Mat1: pMat;
  Vec1: pVn;
  line: Integer;

Procedure adj_set_s(VAR iadj:pVmadj; maxadj:INTEGER; VAR nadj:INTEGER; n:INTEGER;
            VAR xadj:pVinp); Forward;
Procedure adj_print(VAR iadj:pVmadj; nadj:INTEGER; n:INTEGER; VAR xadj:pVinp); Forward;
Procedure shomat  (VAR iadj:pVmadj; VAR iband,ienv:INTEGER; VAR invprm:pVin;
            maxadj:INTEGER; n:INTEGER; VAR perm:pVin; VAR xadj:pVinp); Forward;
Procedure genqmd  (n:INTEGER;VAR xadj:pVinp;VAR iadj:pVmadj; VAR perm:pVin;
            VAR invprm:pVin; VAR marker:pVin; VAR rchset:pVin; VAR nbrhd:pVin;
            VAR qsize:pVin; VAR qlink:pVin; VAR nofsub:INTEGER); Forward;
Procedure perm_inverse(n:INTEGER; VAR perm:pVin; VAR invprm:pVin); Forward;
Procedure smb_factor(n:INTEGER; VAR xadj:pVinp; VAR iadj:pVmadj; VAR perm:pVin;
            VAR invprm:pVin; VAR ixlnz:pVinp; VAR nofnz:INTEGER; VAR xnzsub:pVi99;
            VAR nzsub:pnzsub; VAR maxsub:INTEGER; VAR rchlnk,mrglnk:pVin); Forward;
Procedure setsys  (VAR diag:pVn; VAR env:pVenv; VAR iadj:pVmadj; VAR invprm:pVin;
            VAR xadj:pVinp; maxadj,maxenv,n:INTEGER; VAR rhs:pVn; VAR nzsub:pnzsub;
            VAR xnzsub,lnz:pVi99; VAR xlnz:pVinp); Forward;
Procedure gs_factor(n:INTEGER; VAR ixlnz:pVinp; VAR lnz,xnzsub:pVi99; VAR nzsub:pnzsub;
            VAR diag:pVn; VAR link,first:pVin); Forward;
Procedure gs_solve (n:INTEGER; VAR ixlnz:pVinp; VAR lnz,xnzsub:pVi99; VAR nzsub:pnzsub;
            VAR diag:pVn;VAR rhs:pVn); Forward;
Procedure perm_rv  (n:INTEGER; VAR rhs:pVn; VAR perm:pVin); Forward;

Procedure Solve_sparse_system;

Var
  iadj, iadj2: pVmadj;
  diag,rhs  : pVn;
  env       : pVenv;
  first,invprm,link,marker,mrglnk,nbrhd,perm: pVin;
  i,iband,ienv,maxsub,nadj,nofnz,nofsub: INTEGER;
  lnz,xnzsub: pVi99;
  nzsub:pnzsub;
  qlink,qsize,rchlnk,rchset: pVin;
  xadj,xlnz:pVinp;

Begin

  New(iadj);  New(iadj2);  New(diag);  New(rhs);  New(env);
  New(first);  New(invprm); New(link);  New(marker);
  New(mrglnk); New(nbrhd);  New(perm);
  New(lnz);  New(xnzsub); New(nzsub);
  New(qlink);  New(qsize);  New(rchlnk); New(rchset);
  New(xadj);  New(xlnz);
  New(lnzr);
```

```
  writeln(fp2);
  writeln(fp2,'Penyelesaian sistem persamaan linier menggunakan Algoritma Derajat Minimum.');
  writeln(fp2);
  for i := 1 to n do
  begin
    perm^[i] := i;
    invprm^[i] := i
  end;
  adj_set_s(iadj, maxadj, nadj, n, xadj);
  writeln(fp2);
  writeln(fp2,' Ada NADJ = ',nadj,' adjacency.');
  adj_print(iadj, maxadj, n, xadj);
  shomat(iadj, iband, ienv, invprm, maxadj, n, perm, xadj);
  for i:=1 to maxadj do iadj2^[i] := iadj^[i];  {save iadj in iadj2}
  genqmd(n, xadj, iadj2, perm, invprm, marker, rchset, nbrhd, qsize, qlink, nofsub);
  perm_inverse(n, perm, invprm);
  for i := 1 to n do
  writeln(fp2,i:6, perm^[i]:6, invprm^[i]:6);
  maxsub := maxnzsub;
  smb_factor(n, xadj, iadj, perm, invprm, xlnz, nofnz, xnzsub, nzsub, maxsub, rchlnk, mrglnk);
  setsys(diag, env, iadj, invprm, xadj, maxadj, maxenv, n, rhs, nzsub, xnzsub, lnz, xlnz);
  Dispose(Mat1); Dispose(Vec1);
  gs_factor(n, xlnz, lnz, xnzsub, nzsub, diag, link, first);
  gs_solve(n, xlnz, lnz, xnzsub, nzsub, diag, rhs);
  perm_rv(n, rhs, perm);
  writeln(fp2);
  writeln(fp2,' Solusi :');
  line:=1;
  for i:=1 to n do
  begin
    write(fp2,rhs^[i]:14:6); Inc(line);
    if line MOD 6 = 0 then
    begin
      line:=1; writeln(fp2)
    end
  end;
  writeln(fp2);
  writeln(fp2);
  Dispose(iadj); Dispose(iadj2); Dispose(diag); Dispose(rhs); Dispose(env);
  Dispose(first); Dispose(invprm); Dispose(link); Dispose(marker);
  Dispose(mrglnk); Dispose(nbrhd); Dispose(perm);
  Dispose(lnz); Dispose(xnzsub); Dispose(nzsub);
  Dispose(qlink); Dispose(qsize); Dispose(rchlnk); Dispose(rchset);
  Dispose(xadj); Dispose(xlnz);
  Dispose(lnzr);
end; {Solve_sparse_system}
```

```
Procedure addrhs(VAR invprm:pVin; isub, n:INTEGER; VAR rhs:pVn; value:REAL); Forward;
Procedure addcom(isub,jsub:INTEGER; value:REAL; VAR invprm:pVin; VAR diag:pVn;
          VAR lnz:pVi99; VAR ixlnz:pVinp; VAR nzsub:pnzsub;
          VAR xnzsub:pVi99; n:INTEGER); Forward;

Procedure setsys  (VAR diag:pVn; VAR env:pVenv; VAR iadj:pVmadj; VAR invprm:pVin;
          VAR xadj:pVinp; maxadj,maxenv,n:INTEGER; VAR rhs:pVn; VAR nzsub:pnzsub;
          VAR xnzsub,lnz:pVi99; VAR xlnz:pVinp);
Var     i,isub,j,jsub:INTEGER;
        value:REAL;

Begin

 for i:=1 to n do
 begin
  rhs^[i] := 0.0;
  diag^[i] := 0.0
 end;
 for i:=1 to maxenv do env^[i] := 0.0;
 for i:=1 to n do
  if Vec1^[i]<>0.0 then addrhs(invprm, i, n, rhs, Vec1^[i]);
 for i:=1 to n do diag^[i] := Mat1^[i,i];
 for i := 1 to n do
 begin
  isub := i;
  for j := xadj^[i] Downto xadj^[i+1] do
  begin
   jsub := iadj^[j];
   value := Mat1^[isub,jsub];
   addcom(isub, jsub, value, invprm, diag, lnz, xlnz, nzsub, xnzsub, n)
  end
 end
End; {setsy3}

Procedure addrhs(VAR invprm:pVin; isub, n:INTEGER; VAR rhs:pVn; value:REAL);

Var     i:INTEGER;

Begin
 i := invprm^[isub];
 if (1 <= i) and (i <= n) then
  rhs^[i] := rhs^[i] + value
End; {addrhs}

Procedure addcom(isub,jsub:INTEGER; value:REAL; VAR invprm:pVin; VAR diag:pVn;
          VAR lnz:pVi99; VAR ixlnz:pVinp; VAR nzsub:pnzsub;
          VAR xnzsub:pVi99; n:INTEGER);
Label   fin;
Var     i,j,k,kstrt,kstop,ksub:INTEGER;
```

```
Begin

  i := invprm^[isub];
  j := invprm^[jsub];
  if i=j then
  begin
    diag^[i] := diag^[i] + value;
    goto fin
  end;
  if i<j then goto fin;
  kstrt := ixlnz^[j];
  kstop := ixlnz^[j+1]-1;
  if kstop < kstrt then
  begin
    writeln(fp2);
    writeln(fp2,'ADDCOM - Fatal error');
    writeln(fp2,'  Array IXLNZ salah !');
    writeln(fp2,'  IXLNZ(J) =     ', ixlnz^[j]);
    writeln(fp2,'  IXLNZ(J+1)-1 = ', ixlnz^[j+1]-1);
    Close(fp2);
    Halt(1)
  end;
  ksub := xnzsub^[j];
  for k := kstrt to kstop do
  begin
    if nzsub^[ksub]=i then
    begin
      ixlnz^[k] := ixlnz^[k] + Round(value);
      goto fin
    end;
    Inc(ksub)
  end;
  writeln(fp2);
  writeln(fp2,'ADDCOM - Fatal error');
  writeln(fp2,'  Tidak ada tempat penyimpanan yang disediakan selain untuk elemen ISUB, JSUB');
  writeln(fp2,'  dimana ISUB = ', isub, ' and JSUB = ', jsub);
  Close(fp2);
  Halt(1);
fin:End; {addcom}

Procedure adj_set(VAR iadj:pVmadj; irow, jcol, maxadj:INTEGER; VAR nadj:INTEGER;
                  n:INTEGER; VAR xadj:pVinp); Forward;

Procedure adj_set_s(VAR iadj:pVmadj; maxadj:INTEGER; VAR nadj:INTEGER; n:INTEGER;
                  VAR xadj:pVinp);
Type    plist    = ^List;
        List     = Array[1..100] of Integer;
Var
  i,j,nonz:INTEGER;
  ilist,jlist:plist;
```

```
Begin

 nonz:=0;
 New(ilist); New(jlist);
 ilist^[1]:=0; jlist^[1]:=0;
 for i:=1 to n do
  for j:=1 to n do
   if (i<>j) and (Mat1^[i,j]<>0.0) then
   begin
    Inc(nonz);
    ilist^[nonz+1]:=i; jlist^[nonz+1]:=j
   end;
 Inc(nonz);
 for i := 1 to nonz do
  adj_set(iadj, ilist^[i], jlist^[i], maxadj, nadj, n, xadj);
 Dispose(ilist); Dispose(jlist)
End; {adj_set_3}

 Function Min(a,b:INTEGER):INTEGER;
 begin
  if a<=b then Min:=a else Min:=b
 end;
 Function Max(a,b:INTEGER):INTEGER;
 begin
  if a>=b then Max:=a else Max:=b
 end;

Procedure adj_print(VAR iadj:pVmadj; nadj:INTEGER; n:INTEGER; VAR xadj:pVinp);

Var     i,j,jlo,jmin,jmax,jhi:INTEGER;

Begin
 writeln(fp2);
 {writeln(fp2,'ADJ_PRINT');}
 writeln(fp2,' Tampilan struktur adjacency dari matriks sparse.');
 writeln(fp2,' Total ada ',nadj,' entri.');
 writeln(fp2);
 writeln(fp2,'  Baris       Tidak nol');
 writeln(fp2,' -------------------------');
 for i := 1 to n do
 begin
  jmin := xadj^[i];
  jmax := xadj^[i+1]-1;
  jlo:=jmin;
  while jlo<=jmax do
  begin
   jhi := Min(jlo+9,jmax);
   if jlo=jmin then
   begin
    write(fp2,i:6,'     '); for j:=jlo to jhi do write(fp2,iadj^[j]:6);
```

```
      writeln(fp2)
    end
    else
    begin
      for j:=jlo to jhi do write(fp2,'          ',iadj^[j]:6);
      writeln(fp2)
    end;
    Inc(jlo,10)
  end {while}
 end
End; {adj_print}

Procedure adj_set(VAR iadj:pVmadj; irow, jcol, maxadj:INTEGER; VAR nadj:INTEGER;
                  n:INTEGER; VAR xadj:pVinp);
Label   20,fin;
Var     i,j,k,kback,klo,khi:INTEGER;

Begin
 if (irow=0) or (jcol=0) then
 begin
  writeln(fp2);
  writeln(fp2,' Jumlah persamaan N = ', n);
  nadj := 0;
  for i:=1 to n+1 do xadj^[i] := 1;
  for i:=1 to maxadj do iadj^[i] := 0;
  goto fin
 end;
 if irow=jcol then goto fin;
 if xadj^[n+1] > maxadj + 1 then
 begin
  writeln(fp2);
  writeln(fp1,'ADJ_SET - Fatal error');
  writeln(fp1,' Semua tempat penyimpanan yang tersedia telah digunakan.');
  writeln(fp1,' Tidak ada lagi informasi yang dapat disimpan.');
  writeln(fp1,' Kesalahan terjadi untuk ');
  writeln(fp1,' Baris IROW = ',irow);
  writeln(fp1,' Kolom JCOL = ',jcol);
  Close(fp2);
  Halt(2)
 end;
 if irow > n then
 begin
  writeln(fp2);
  writeln(fp2,'ADJ_SET - Fatal error');
  writeln(fp2,'  IROW > N.');
  writeln(fp2,'  IROW = ', irow);
  writeln(fp2,'  N = ', n);
  Close(fp2);
  Halt(2)
 end
```

```
  else if irow < 1 then
  begin
    writeln(fp2);
    writeln(fp2,'ADJ_SET - Fatal error');
    writeln(fp2,'  IROW < 1');
    writeln(fp2,'  IROW = ', irow);
    Close(fp2);
    Halt(2)
  end
  else if jcol > n then
  begin
    writeln(fp2);
    writeln(fp2,'ADJ_SET - Fatal error');
    writeln(fp2,'  JCOL > N.');
    writeln(fp2,'  JCOL = ',jcol);
    writeln(fp2,'  N = ', n);
    Close(fp2);
    Halt(2)
  end
  else if jcol < 1 then
  begin
    writeln(fp2);
    writeln(fp2,'ADJ_SET - Fatal error');
    writeln(fp2,'  JCOL < 1');
    writeln(fp2,'  JCOL = ', jcol);
    Close(fp2);
    Halt(2)
  end;
  i := irow;
  j := jcol;
 20:klo := xadj^[i];
  khi := xadj^[i+1]-1;
  for k := klo to khi do
  begin
    if iadj^[k]=j then
    begin
      if  i=irow then
      begin
        i := jcol;
        j := irow;
        goto 20
      end;
      goto fin
    end
  end;
  for k := xadj^[i+1] to xadj^[n+1] do
  begin
    kback := xadj^[n+1] + xadj^[i+1] - k;
    iadj^[kback+1] := iadj^[kback]
  end;
```

```
  iadj^[xadj^[i+1]] := j;
  for k := i+1 to n+1 do xadj^[k] := xadj^[k] + 1;
  nadj := xadj^[n+1] - 1;
  if i=irow then
  begin
    i := jcol;
    j := irow;
    goto 20
  end;
fin:End; {adj_set}

Procedure shomat(VAR iadj:pVmadj; VAR iband,ienv:INTEGER; VAR invprm:pVin;
                 maxadj:INTEGER; n:INTEGER; VAR perm:pVin; VAR xadj:pVinp);
Var      i,iadd,itemp,j,jlo,jhi,k,nonz:INTEGER;
         band:STRING[100];
Begin
  iband := 0;
  ienv := 0;
  nonz := 0;
  if n > 100 then
  begin
    writeln(fp2);
    writeln(fp2,'SHOMAT - Fatal error');
    writeln(fp2,'  N terlalu besar !');
    writeln(fp2,'  Maksimal n adalah 100.');
    writeln(fp2,'  Input Anda adalah ', n);
    Close(fp2);
    Halt(3)
  end;
  writeln(fp2);
  writeln(fp2,' Tampilan struktur tidak nol dari matriks.');
  writeln(fp2);
    for i := 1 to n do
    begin
      for k := 1 to n do band[k] := ' ';
      band[i] := 'X';
      iadd := 0;
      jlo := xadj^[perm^[i]];
      jhi := xadj^[perm^[i]+1]-1;
      for j := jlo to jhi do
      begin
        itemp := invprm^[iadj^[j]];
        if itemp < i then Inc(nonz);
        iband := Max(iband,i-itemp);
        band[itemp] := 'X';
        iadd := Max(iadd,i-itemp)
      end;
      write(fp2,i:6,' '); for j:=1 to n do write(fp2,band[j]);
      writeln(fp2);
      ienv := ienv + iadd
```

```
   end;
  writeln(fp2);
End; {shomat}

Procedure qmdrch(root:INTEGER; VAR xadj:pVinp; VAR iadj:pVmadj; VAR deg, marker:pVin;
          VAR rchsze:INTEGER;VAR rchset:pVin;VAR nhdsze:INTEGER; VAR
nbrhd:pVin;n:INTEGER);
          Forward;
Procedure qmdupd(VAR xadj:pVinp; VAR iadj:pVmadj; VAR nlist:INTEGER; VAR list:pVin;
          VAR deg,qsize,qlink,marker,rchset,nbrhd:pVin; n:INTEGER);
          Forward;
Procedure qmdqt (root:INTEGER; VAR xadj:pVinp; VAR iadj:pVmadj; VAR marker:pVin;
          VAR rchsze:INTEGER; VAR rchset,nbrhd:pVin; n:INTEGER); Forward;


Procedure genqmd(n:INTEGER;VAR xadj:pVinp;VAR iadj:pVmadj; VAR perm:pVin;
          VAR invprm:pVin; VAR marker:pVin; VAR rchset:pVin; VAR nbrhd:pVin;
          VAR qsize:pVin; VAR qlink:pVin; VAR nofsub:INTEGER);
Label 20,30,50,60,80;

Var       j,mindeg,ndeg,nhdsze,node,num,nump1,nxnode,rchsze,search,thresh:INTEGER;
          i,inode,ip,irch,np:INTEGER;
          deg,temp,temp1: pVin;

Begin

  New(deg); New(temp); New(temp1);
  mindeg := n;
  nofsub := 0;
  for node := 1 to n do
  begin
    perm^[node] := node;
    invprm^[node] := node;
    marker^[node] := 0;
    qsize^[node] := 1;
    qlink^[node] := 0;
    ndeg := xadj^[node+1]-xadj^[node];
    deg^[node] := ndeg;
    mindeg := Min(mindeg,ndeg)
  end;
  num := 0;
20:search := 1;
  thresh := mindeg;
  mindeg := n;
30:nump1 := num+1;
  search := Max(search,nump1);
  for j := search to n do
  begin
    node := perm^[j];
    if marker^[node] >= 0 then
```

```
    begin
      ndeg := deg^[node];
      if ndeg <= thresh then goto 50;
      mindeg := Min(mindeg,ndeg)
    end
  end;
  goto 20;
50:search := j;
  nofsub := nofsub+deg^[node];
  marker^[node] := 1;
  qmdrch(node, xadj, iadj, deg, marker, rchsze, rchset, nhdsze, nbrhd, n);
  nxnode := node;
60:Inc(num);
  np := invprm^[nxnode];
  ip := perm^[num];
  perm^[np] := ip;
  invprm^[ip] := np;
  perm^[num] := nxnode;
  invprm^[nxnode] := num;
  deg^[nxnode] := -1;
  nxnode := qlink^[nxnode];
  if nxnode > 0 then goto 60;
  if rchsze <= 0 then goto 80;
  for i:=1 to n do
  begin
    temp^[i]:=0;
    temp1^[i]:=0
  end;
  for i:=rchsze to n do temp^[i-rchsze+1]:=rchset^[i+1];
  for i:=nhdsze to n do temp1^[i-nhdsze+1]:=nbrhd^[i+1];
  qmdupd(xadj,iadj,rchsze,rchset,deg,qsize,qlink,marker,temp,temp1,n);
  for i:=rchsze to n do rchset^[i+1]:=temp^[i-rchsze+1];
  for i:=nhdsze to n do nbrhd^[i+1]:=temp1^[i-nhdsze+1];
  marker^[node] := 0;
  for irch := 1 to rchsze do
  begin
    inode := rchset^[irch];
    if marker^[inode] >= 0 then
    begin
      marker^[inode] := 0;
      ndeg := deg^[inode];
      mindeg := Min(mindeg,ndeg);
      if ndeg <= thresh then
      begin
        mindeg := thresh;
        thresh := ndeg;
        search := invprm^[inode]
      end
    end
  end;
```

```
 if nhdsze > 0 then
   qmdqt(node, xadj, iadj, marker, rchsze, rchset, nbrhd, n);
80:if num < n then goto 30;
 Dispose(deg); Dispose(temp); Dispose(temp1)
End; {genqmd}

Procedure qmdmrg(VAR xadj:pVinp;VAR iadj:pVmadj;VAR deg,qsize,qlink,marker:pVin;
          deg0,nhdsze:INTEGER; VAR nbrhd,rchset,ovrlp:pVin;n:INTEGER); Forward;

Procedure qmdupd(VAR xadj:pVinp; VAR iadj:pVmadj; VAR nlist:INTEGER; VAR list:pVin;
          VAR deg,qsize,qlink,marker,rchset,nbrhd:pVin; n:INTEGER);
Label fin;
Var     deg0,deg1,il,inode,irch,inhd,j,jstrt,jstop,mark,nabor,nhdsze,node,rchsze:INTEGER;

Begin

 if nlist <= 0 then goto fin;
 deg0 := 0;
 nhdsze := 0;
 for il := 1 to nlist do
 begin
  node := list^[il];
  deg0 := deg0+qsize^[node];
  jstrt := xadj^[node];
  jstop := xadj^[node+1]-1;
  for j := jstrt to jstop do
  begin
   nabor := iadj^[j];
   if (marker^[nabor]=0) and (deg^[nabor] < 0) then
   begin
    marker^[nabor] := -1;
    Inc(nhdsze);
    nbrhd^[nhdsze] := nabor
   end
  end
end;
if nhdsze > 0 then
  qmdmrg(xadj,iadj,deg,qsize,qlink,marker,deg0,nhdsze,nbrhd,rchset,nbrhd,n);
for il := 1 to nlist do
begin
 node := list^[il];
 mark := marker^[node];
 if (mark=0) or (mark=1) then
 begin
  marker^[node] := 2;
  qmdrch(node, xadj, iadj, deg, marker, rchsze, rchset, nhdsze, nbrhd, n);
  deg1 := deg0;
  for irch := 1 to rchsze do
  begin
   inode := rchset^[irch];
```

```
      deg1 := deg1 + qsize^[inode];
      marker^[inode] := 0
    end;
    deg^[node] := deg1-1;
    for inhd := 1 to nhdsze do
    begin
     inode := nbrhd^[inhd];
     marker^[inode] := 0
    end
   end
 end;
fin:End; {qmdupd}

Procedure qmdmrg(VAR xadj:pVinp;VAR iadj:pVmadj;VAR deg,qsize,qlink,marker:pVin;
          deg0,nhdsze:INTEGER; VAR nbrhd,rchset,ovrlp:pVin;n:INTEGER);
Label   20,70,90,110,fin;
Var     deg1,inhd,j,jstrt,jstop,mark,nabor,novrlp,rchsze,root:INTEGER;
        iov,irch,head,link,lnode,mrgsze,node:INTEGER;

Begin
 if nhdsze <= 0 then goto fin;
 for inhd := 1 to nhdsze do
 begin
  root := nbrhd^[inhd];
  marker^[root] := 0
 end;
 for inhd := 1 to nhdsze do
 begin
  root := nbrhd^[inhd];
  marker^[root] := - 1;
  rchsze := 0;
  novrlp := 0;
  deg1 := 0;
20: jstrt := xadj^[root];
  jstop := xadj^[root+1]-1;
  for j := jstrt to jstop do
  begin
   nabor := iadj^[j];
   root := -nabor;
   if nabor < 0 then goto 20;
   if nabor=0 then goto 70;
   mark := marker^[nabor];
   if mark=0 then
   begin
    Inc(rchsze);
    rchset^[rchsze] := nabor;
    deg1 := deg1+qsize^[nabor];
    marker^[nabor]:= 1
   end
   else if mark=1 then
```

```
      begin
        Inc(novrlp);
        ovrlp^[novrlp] := nabor;
        marker^[nabor] := 2
      end
    end; {j loop}
70: head := 0;
    mrgsze := 0;
    for iov := 1 to novrlp do
    begin
      node := ovrlp^[iov];
      jstrt := xadj^[node];
      jstop := xadj^[node+1]-1;
      for j := jstrt to jstop do
      begin
        nabor := iadj^[j];
        if marker^[nabor]=0 then
        begin
          marker^[node] := 1;
          goto 110
        end
      end;
      mrgsze := mrgsze + qsize^[node];
      marker^[node] := -1;
      lnode := node;
90:   link := qlink^[lnode];
      if link > 0 then
      begin
        lnode := link;
        goto 90
      end;
      qlink^[lnode] := head;
      head := node;
110:end; {iov loop}
    if head > 0 then
    begin
      qsize^[head] := mrgsze;
      deg^[head] := deg0+deg1-1;
      marker^[head] := 2
    end;
    root := nbrhd^[inhd];
    marker^[root] := 0;
    for irch := 1 to rchsze do
    begin
      node := rchset^[irch];
      marker^[node] := 0
    end
  end; {inhd loop}
fin:End; {qmdmrg}
```

```
Procedure qmdqt(root:INTEGER; VAR xadj:pVinp; VAR iadj:pVmadj; VAR marker:pVin;
          VAR rchsze:INTEGER; VAR rchset,nbrhd:pVin; n:INTEGER);
Label   10,40,60;
Var     inhd,irch,j,jstrt,jstop,link,nabor,node:INTEGER;

Begin
 irch := 0;
 inhd := 0;
 node := root;
10:jstrt := xadj^[node];
 jstop := xadj^[node+1]-2;
 for j := jstrt to jstop do
 begin
  Inc(irch);
  iadj^[j] := rchset^[irch];
  if irch >= rchsze then goto 40
 end;
 link := iadj^[jstop+1];
 node := -link;
 if link >= 0 then
 begin
  Inc(inhd);
  node := nbrhd^[inhd];
  iadj^[jstop+1] := -node
 end;
 goto 10;
40:iadj^[j+1] := 0;
 for irch := 1 to rchsze do
 begin
  node := rchset^[irch];
  if marker^[node] >= 0 then
  begin
   jstrt := xadj^[node];
   jstop := xadj^[node+1]-1;
   for j := jstrt to jstop do
   begin
    nabor := iadj^[j];
    if marker^[nabor] < 0 then
    begin
     iadj^[j] := root;
     goto 60
    end
   end
  end;
60:end {irch loop}
End; {qmdqt}

Procedure qmdrch(root:INTEGER; VAR xadj:pVinp; VAR iadj:pVmadj; VAR deg, marker:pVin;
                 VAR rchsze:INTEGER;VAR rchset:pVin;VAR nhdsze:INTEGER;
                 VAR nbrhd:pVin;n:INTEGER);
```

```
Label   20,50,fin;
Var     i,istrt,istop,j,jstrt,jstop,nabor,node:INTEGER;

Begin

  nhdsze := 0;
  rchsze := 0;
  istrt := xadj^[root];
  istop := xadj^[root+1]-1;
  for i := istrt to istop do
  begin
    nabor := iadj^[i];
    if nabor=0 then goto fin;
    if marker^[nabor]<>0 then goto 50;
    if deg^[nabor] >= 0 then
    begin
      Inc(rchsze);
      rchset^[rchsze] := nabor;
      marker^[nabor] := 1;
      goto 50
    end;
    marker^[nabor] := -1;
    nhdsze := nhdsze+1;
    nbrhd^[nhdsze] := nabor;
20: jstrt := xadj^[nabor];
    jstop := xadj^[nabor+1]-1;
    for j := jstrt to jstop do
    begin
      node := iadj^[j];
      nabor := -node;
      if node < 0 then goto 20;
      if node = 0 then goto 50;
      if marker^[node]=0 then
      begin
        Inc(rchsze);
        rchset^[rchsze] := node;
        marker^[node] := 1
      end
    end;
50:end; {i loop}
fin:End; {qmd_rch}

Procedure perm_inverse(n:INTEGER; VAR perm:pVin; VAR invprm:pVin);

Var     i,k:INTEGER;

Begin
  for i := 1 to n do
  begin
    k := perm^[i];
```

```
    invprm^[k] := i
  end
End;

Procedure smb_factor(n:INTEGER; VAR xadj:pVinp; VAR iadj:pVmadj; VAR perm:pVin;
                     VAR invprm:pVin; VAR ixlnz:pVinp; VAR nofnz:INTEGER; VAR xnzsub:pVi99;
                     VAR nzsub:pnzsub; VAR maxsub:INTEGER; VAR rchlnk,mrglnk:pVin);
Label   20,40,50,70,90,110,130,150,160;
Var     k,knz,mrgk,mrkflg,nabor,node,np1,nzbeg,nzend:INTEGER;
        i,inz,j,jstrt,jstop,kxsub,lmax,m,rchm:INTEGER;
        marker:pVin;

Begin

  New(marker);
  nzbeg := 1;
  nzend := 0;
  ixlnz^[1] := 1;
  for k := 1 to n do mrglnk^[k] := 0;
  for k:=1 to n do marker^[k] := 0;
  np1 := n + 1;
  for k := 1 to n do
  begin
    knz := 0;
    mrgk := mrglnk^[k];
    mrkflg := 0;
    marker^[k] := k;
    if mrgk <> 0 then marker^[k] := marker^[mrgk];
    xnzsub^[k] := nzend;
    node := perm^[k];
    jstrt := xadj^[node];
    jstop := xadj^[node+1]-1;
    if jstrt > jstop then goto 160;
    rchlnk^[k] := np1;
    for j := jstrt to jstop do
    begin
      nabor := iadj^[j];
      nabor := invprm^[nabor];
      if nabor > k then
      begin
        rchm := k;
20:     m := rchm;
        rchm := rchlnk^[m];
        if rchm <= nabor then goto 20;
        Inc(knz);
        rchlnk^[m] := nabor;
        rchlnk^[nabor] := rchm;
        if marker^[nabor] <> marker^[k] then mrkflg := 1
      end
    end; {j loop}
```

```
    lmax := 0;
    if (mrkflg <> 0) or (mrgk = 0) then goto 40;
    if mrglnk^[mrgk] <> 0 then goto 40;
    xnzsub^[k] := xnzsub^[mrgk] + 1;
    knz := ixlnz^[mrgk+1] - (ixlnz^[mrgk]+1);
    goto 150;
40: i:=k;
50: i := mrglnk^[i];
    if i = 0 then goto 90;
    inz := ixlnz^[i+1]-(ixlnz^[i]+1);
    jstrt := xnzsub^[i]+1;
    jstop := xnzsub^[i]+inz;
    if inz > lmax then
    begin
      lmax := inz;
      xnzsub^[k] := jstrt
    end;
    rchm := k;
    for j := jstrt to jstop do
    begin
      nabor := nzsub^[j];
70:   m := rchm;
      rchm := rchlnk^[m];
      if rchm < nabor then goto 70;
      if rchm <> nabor then
      begin
        Inc(knz);
        rchlnk^[m] := nabor;
        rchlnk^[nabor] := rchm;
        rchm := nabor
      end
    end;
    goto 50;
90: if knz = lmax then goto 150;
    if nzbeg > nzend then goto 130;
    i := rchlnk^[k];
    for jstrt := nzbeg to nzend do
    begin
      if nzsub^[jstrt] = i then goto 110;
      if nzsub^[jstrt] > i then goto 130
    end;
    goto 130;
110:xnzsub^[k] := jstrt;
    for j := jstrt to nzend do
    begin
      if nzsub^[j] <> i then goto 130;
      i := rchlnk^[i];
      if i > n then goto 150
    end;
    nzend := jstrt-1;
```

```
130:nzbeg := nzend+1;
   nzend := nzend+knz;
   if nzend > maxsub then
   begin
    writeln(fp2);
    writeln(fp2,'SMB_FACTOR - Fatal error');
    writeln(fp2,' Tempat penyimpanan untuk elemen tidak tidak mencukupi.');
    writeln(fp2,' MAXSUB := ', maxsub);
    writeln(fp2,' Melebihi NZEND := ', nzend);
    Close(fp2);
    Halt(4)
   end;
   i := k;
   for j := nzbeg to nzend do
   begin
    i := rchlnk^[i];
    nzsub^[j] := i;
    marker^[i] := k
   end;
   xnzsub^[k] := nzbeg;
   marker^[k] := k;
150:if knz > 1 then
   begin
    kxsub := xnzsub^[k];
    i := nzsub^[kxsub];
    mrglnk^[k] := mrglnk^[i];
    mrglnk^[i] := k
   end;
160:ixlnz^[k+1] := ixlnz^[k] + knz
 end; {k loop}
 nofnz := ixlnz^[n] - 1;
 maxsub := xnzsub^[n];
 xnzsub^[n+1] := xnzsub^[n];
 Dispose(marker)
end; {smb_factor}

Procedure gs_factor(n:INTEGER; VAR ixlnz:pVinp; VAR lnz,xnzsub:pVi99; VAR nzsub:pnzsub;
            VAR diag:pVn; VAR link,first:pVin);
Label   20,40;
Var     i,ii,istrt,istop,isub,j,k,kfirst,Disposek:INTEGER;
        diagj,ljk:REAL;
        temp :pVn;

Begin

 New(temp);
 for i:=1 to n do
 begin
  link^[i] := 0;
  temp^[i] := 0.0
```

```
  end;
  for i:=1 to n do lnzr^[i]:=-1.0;
  for i:=n+1 to 99 do lnzr^[i]:=0.0;
  for j := 1 to n do
  begin
   diagj := 0.0;
   Disposek := link^[j];
20: k := Disposek;
   if k = 0 then goto 40;
   Disposek := link^[k];
   kfirst := first^[k];
   ljk := lnzr^[kfirst];
   diagj := diagj + ljk*ljk;
   istrt := kfirst+1;
   istop := ixlnz^[k+1]-1;
   if istop < istrt then goto 20;
   first^[k] := istrt;
   i := xnzsub^[k] + (kfirst - ixlnz^[k]) + 1;
   isub := nzsub^[i];
   link^[k] := link^[isub];
   link^[isub] := k;
   for ii := istrt to istop do
   begin
    isub := nzsub^[i];
    temp^[isub] := temp^[isub] + lnzr^[ii] * ljk;
    Inc(i)
   end;
   goto 20;
40: diagj := diag^[j] - diagj;
   if diagj <= 0.0 then
   begin
    writeln(fp2);
    writeln(fp2,'GSFCT - Fatal error');
    writeln(fp2,'  Elemen diagonal nol atau negatif !');
    writeln(fp2,'  DIAG(J) := ', diagj);
    writeln(fp2,'  for diagonal J := ', j);
    writeln(fp2);
    writeln;
    writeln;
    writeln;
    writeln(' ========== E R R O R !! ========== ');
    writeln;
    writeln('   Matriks tidak definit positif !');
    writeln('   Silahkan cek data matriks Anda.');
    writeln;
    writeln(' ========== E R R O R !! ========== ');
    writeln;
    writeln;
    writeln(' Tekan sembarang tombol untuk menutup jendela ini...');
    Readkey;
```

```
      DoneWinCrt
    end;
    diagj := sqrt(diagj);
    diag^[j] := diagj;
    istrt := ixlnz^[j];
    istop := ixlnz^[j+1] - 1;
    if istop >= istrt then
    begin
     first^[j] := istrt;
     i := xnzsub^[j];
     isub := nzsub^[i];
     link^[j] := link^[isub];
     link^[isub] := j;
     for ii:=istrt to istop do
     begin
       isub := nzsub^[i];
       lnzr^[ii] := (lnzr^[ii] - temp^[isub]) / diagj;
       temp^[isub] := 0.0;
       Inc(i)
     end
    end
  end; {j loop}
  Dispose(temp)
End; {gs_factor}

Procedure gs_solve (n:INTEGER; VAR ixlnz:pVinp; VAR lnz,xnzsub:pVi99; VAR nzsub:pnzsub;
                    VAR diag:pVn;VAR rhs:pVn);
Var     i,ii,istrt,istop,isub,j,jj:INTEGER;
        rhsj,s: REAL;

Begin

 for j := 1 to n do
 begin
   rhsj := rhs^[j] / diag^[j];
   rhs^[j] := rhsj;
   istrt := ixlnz^[j];
   istop := ixlnz^[j+1] - 1;
   i := xnzsub^[j];
   for ii := istrt to istop do
   begin
     isub := nzsub^[i];
     rhs^[isub] := rhs^[isub] - lnzr^[ii] * rhsj;
     Inc(i)
   end
 end;
 j := n;
 for jj := 1 to n do
 begin
   s := rhs^[j];
```

```
    istrt := ixlnz^[j];
    istop := ixlnz^[j+1] - 1;
    i := xnzsub^[j];
    for ii := istrt to istop do
    begin
      isub := nzsub^[i];
      s := s - lnzr^[ii] * rhs^[isub];
      Inc(i)
    end;
    rhs^[j] := s / diag^[j];
    Dec(j)
  end
End; {gs_solve}

Procedure perm_rv(n:INTEGER; VAR rhs:pVn; VAR perm:pVin);

Label   20,30,40,fin;
Var     i,iput,istart:INTEGER;
        pull,put: REAL;

Begin

  for i := 1 to n do perm^[i] := -perm^[i];
  istart := 0;
20:Inc(istart);
  if istart > n then goto fin;
  if perm^[istart] > 0 then goto 20;
  if abs(perm^[istart] ) <> istart then goto 30;
  perm^[istart] := abs(perm^[istart]);
  goto 20;
30:perm^[istart] := abs(perm^[istart]);
  iput := istart;
  pull := rhs^[iput];
40:iput := abs(perm^[iput]);
  put := rhs^[iput];
  rhs^[iput] := pull;
  pull := put;
  if perm^[iput] > 0 then goto 20;
  perm^[iput] := abs(perm^[iput]);
  goto 40;
fin:End; {perm_rv}

Procedure ReadSystem(VAR fp:TEXT;VAR N:Integer; VAR Matrix:pMat; VAR Vector:pVn);

Var     i,j:Integer;
        temp:REAL;

Begin
  Read(fp,N);
  for i:=1 to N do
```

```
  begin
    for j:=1 to n do read(fp,Matrix^[i,j]);
    readln(fp,Vector^[i])
  end
End;

Procedure WriteSystem(VAR fp:TEXT; N:Integer; Matrix:pMat; Vector:pVn);
Var     i,j:Integer;
        temp:REAL;
Begin

  writeln(fp,' Sistem linier yang akan diselesaikan :');
  Writeln(fp,' N = ',N);
  for i:=1 to N do
  begin
    for j:=1 to n do write(fp,Matrix^[i,j]:8:3);
    writeln(fp,'  ',Vector^[i]:8:3)
  end;
  writeln(fp)
End;

Function Test_sym:Boolean;
Var i,j:Integer;
Begin
  Test_sym:=TRUE;
  for i:=1 to n do
    for j:=1 to n do
      if Mat1^[i,j]<>Mat1^[j,i] then
      begin
        Test_sym:=FALSE;
        exit
      end
End;
{main program}
BEGIN
  New(Mat1); New(Vec1);
  writeln;
  writeln('*****************************************************************');
  writeln('*                                                               *');
  writeln('*          Program Algoritma Derajat Minimum                    *');
  writeln('*                                                               *');
  writeln('*              Oleh Wiwin Winanto                               *');
  writeln('*                    Oktober 2001                               *');
  writeln('*****************************************************************');
  writeln('                                                                 ');
  writeln('   Catatan :                                                     ');
  writeln('   - data dibaca dari sebuah file,                               ');
  writeln('   - hasil ditempatkan pada sebuah file output,                  ');
  writeln('   - data matriks input harus simetri dan definit positif.       ');
  writeln('=================================================================');
```

```
 writeln;
 writeln;
 write(' Isi nama file input (dengan ekstensi *.dat) : '); read(name);
 {$I-}
 Assign(fp1,name); Reset(fp1);
 {$I+}
 if IOResult<>0 then
 begin
  writeln;
  writeln(' File tidak ditemukan !');
  writeln;
  writeln;
  writeln(' Tekan sembarang tombol untuk menutup jendela ini...');
  Readkey;
  DoneWinCrt
 end;
 ReadSystem(fp1,N,Mat1,Vec1);
 Close(fp1);
 Assign(fp2,'hasil.lst'); Rewrite(fp2);
 writeln(fp2);
 writeln(fp2,'*************************************************************');
 writeln(fp2,'*                                                           *');
 writeln(fp2,'*          Program Algoritma Derajat Minimum                *');
 writeln(fp2,'*                                                           *');
 writeln(fp2,'*              Oleh Wiwin Winanto                           *');
 writeln(fp2,'*                        Oktober 2001                       *');
 writeln(fp2,'*************************************************************');
 writeln(fp2);
 WriteSystem(fp2, N, Mat1, Vec1);
 if Not Test_sym then
 begin
  writeln;
  writeln;
  writeln;
  writeln(' ========== E R R O R !! ========== ');
  writeln;
  writeln('  Sistem persamaan tidak simetri !');
  writeln('  Silahkan cek data matriks Anda.');
  writeln;
  writeln(' ========== E R R O R !! ========== ');
  writeln;
  writeln;
  writeln(' Tekan sembarang tombol untuk menutup jendela ini...');
  Readkey;
  DoneWinCrt
 end;

 Solve_sparse_system;

 writeln(fp2);
```

```
  writeln(fp2,' ADM');
  writeln(fp2,'  Hasil yang benar.');
fin: close(fp2);
  writeln;
  writeln;
  writeln(' Program mendapatkan penyelesaian dari data Anda.');
  writeln(' Hasil penyelesaian dapat dilihat pada file hasil.lst.');
  writeln;
  writeln;
  writeln(' Tekan sembarang tombol untuk menutup jendela ini...');
  Readkey;
  DoneWinCrt
END.

{end of file ADM.pas}
```